A method and system for providing copy-protection on an storage medium by randomizing locations upon write access, and a player and a storage medium for use in such a system.

The invention relates to a method for providing copy-protection to a data storage medium, in particular to solid state memory modules. With advancing technology, next generations of portable audio playback and recording devices will be based on solid state technology. Arguments in favor are based on weight, power and shockproofness considerations.

Software providers, e.g. music publishers, require measures against unauthorized copying of the digitally stored information, with little or preferably no inconvenience to an authorized user. In addition, the method and system should support such business models as rental, try-before-you-buy, and controlled copying (e.g. super distribution). A particular problem is posed by devices that can potentially access all information on the storage medium, without complying with protection standards.

Known anti-copying solutions use a unique identification code (ID) that is 'engraved' in the storage medium. At some point in time, this may be disadvantageous because of privacy considerations. Furthermore, as will be explained below, methods which mainly rely on such an ID do not provide adequate protection against a copying scheme known as a 'replay attack'.

It is therefore an object of the invention to provide a method and system that provides protection against replay attacks, without necessarily employing a unique ID, in a relatively inexpensive manner that requires only moderate processing facilities.

The basic idea for this copy protection method and system is that the data is encrypted using a key that critically depends on the location(s) in which the data is stored, and which is combined with a method that renders it impossible to predict where the data will be actually stored on the medium. Accordingly, copying of the data will result in an unpredictable change of the storage location, thus breaking the critical relation between the latter and the encryption key. Therefore, once the data has been moved, it can never be recovered, provided that the cryptography is sufficiently strong, the random number generator is cryptographically strong, and any secrets are kept well hidden.

In consequence, amongst other things, it is an object of the present invention to provide an inexpensive method for storing data on storage media, where the relation between encryption key and storage location will be disrupted upon copying operations.

The present invention is particularly suited for solid state memory modules which provide easy random access to any location in the memory, be it on the basis of a bit, a byte or on some other entity such as a uniform sized sector that relates to the access width of the memory in question.

Now, therefore according to one of its aspects the invention is characterized in that the data on the storage medium are encrypted with a key K which depends on the position ($L_1$, $L_2$, $L_3$) of the data on the storage medium, and that in each write operation the data is stored in locations on the storage medium that are chosen at random.

The invention also relates to a system arranged for implementing a method as claimed in claim 1, a player for playing a recording prepared according to a method as claimed in claim 1, and a record carrier prepared according to a method as claimed in claim 1. Further advantageous aspects of the invention are recited in dependent Claims.

These and other objects of the invention will be apparent from and elucidated with reference to the embodiments described hereinafter.

In the drawings:

Figure 1 shows a conceptual two-player arrangement

Figure 2 illustrates the mechanism of 'replay attacks' in the prior art.

Figure 3 shows a schematic diagram of a storage medium embodiment;

Figure 4 shows an example of a file structure

Figures 5A and 5B illustrate an example of a method in accordance with the invention and how this method prevents 'replay attacks'.

Figures 6A and 6B illustrate a further example for a method in accordance with the invention.

Figure 1 illustrates a conceptual two-player arrangement, with two players A and B, and a removable module C that may be transposed between the players. As shown, both players have appropriate means for inserting the module. In the rest of the discussion it is assumed that this removable module may be accessed by other means as well (e.g. PC based readers).

This poses the risk of unauthorized copying of the data on the module, assuming that the players A and B do not allow unauthorized copying. The preferred embodiments are described in relation to a Solid State Audio player and module, although the invention may be used in a broader context.

Within a few years, Solid State Audio (SSA) players are expected to become a new standard for portable audio playback devices. This is mainly due to many advantages on weight, size, power use, and shock resistance, with respect to current solutions using disc or tape. Currently available SSA players combine 32–64 MB of flash memory and audio compression techniques such as MPEG 1 layer III (MP3) or AAC to achieve up to one hour of (near) CD quality music playing time. Due to the digital nature of these devices and the associated ease of copying, however, the music industry insists on proper copyright protection features.

One of the tools for copy protection of digital content is encryption. While encryption by itself does not prevent illegal copying, it does render such copies useless, as the original content can be retrieved only by decrypting it using the proper key. As a result, playback of the content is limited to those devices that have access to that key. It is an objective of the copy protection system to manage the keys in such a way that illegal copying is prevented, while at the same time not inconveniencing legal and intended use of the content.

Most of the memory modules for solid state multimedia storage applications comprise a large flash memory and an on-board controller. The controller may or may not be integrated, and multiple separate memory chips may be employed on the module. Examples of such multimedia memory modules are: Memory Stick (Sony), SmartMedia (SSFDC Forum), Miniature Card (MC Forum), Compact Flash (PCMCIA Forum), Multimedia Card (MMC Association). In addition, these devices can be thought of as block devices, similar to hard disk drives, where memory accesses occur by addressing sectors (typically 512 bytes) on the module. Indeed, some of the modules listed above employ the ATA interface standard, which is used to connect hard disks and other peripherals to a PC. This enables easy duplication (bit by bit) of the content of such memory modules using a PC. Other modules use a proprietary interface and command set, but still are block based, i.e. individual sectors on the module can be addressed and modified.

In the following, it is assumed (see Figure 1) that a SSA player employs detachable memory modules, which can be accessed by other means as well (e.g. PC based readers).

Basically, two approaches exist for copy protection. The first is to bind the audio to a specific player by providing each individual player with a unique, secret, number that is used as the key to encrypt the audio. Therefore, the audio stored on memory modules by one player will play on that player only. Of course, this is very annoying if one has multiple SSA players. It is required that one is able to play music stored on a memory module, regardless of the SSA device used to download it onto the module. What should be prevented, however, is that a user can copy the audio content to another module and be able to play from both.

One known solution is to embed a unique identification code (ID) in the memory module, which can be read by the application, but which can not be changed. This identification code can then be used to generate an encryption key, which is specific for the module.

Another known solution is to make use of defects in the memory modules, which naturally occur as a result of the manufacturing processes used to fabricate cheap but high storage capacity flash memories. The locations of these natural defects probably will be unique for each module, and as such can act as a 'fingerprint' of that device. Again, a unique key can be generated, which is specific for the module.

These known solutions, however, necessitate a unique identification code, and they do not provide protection against replay attacks. A 'replay attack' is a form of copying in which an unauthorized copy is made from one system (system 1) to another (system 2), where the unauthorized (but unplayable copy) on system 2 can be used to restore a playable copy on system over and over again, even after expiration of the original copy. Figure 2 illustrates this in more detail. Each system comprises a unique identification code, represented by ID1 for system 1 and ID2 for system 2, and contains files in which the content is stored as a sequence of separate blocks. In this example the data in respect of rights and usage on the original copy are encrypted with a key that is derived from ID1 and a secret S. In a 'try-before-you-buy' or a rental business model, further access to the data is denied after a certain period of time, or after a number of uses. Copying the data to a system having a unique identification code ID2 (second step in Figure 2) will not render a usable copy, since the identification code does not match the code ID1. However, this copy is exactly (bit-by-bit) the same as the original. It can at any time be recopied back from system 2 to system 1 and that copy of a copy can be used again. This enables a fraudulent customer to retain on system 2 a copy that can be recopied again and again on system 1 where it will be usable. So, after obtaining content on a 'try-before-you-buy' basis, the fraudulent customer copies the data from system 1 to system 2,

and recopies it again and again from system 2 to system 1 in order to keep 'trying'. 'Try-before-you-buy' thus has become 'try-indefinitely.' Likewise this scheme can be used to pay once for a rental and have a copy for ever.

To effectively use a storage device, it is necessary to implement a file system by means of which the user data is organized and accessed. By treating the memory module as a block device, the creation and management of a file system is left to the application. In a PC environment, where the operating system already has built-in file system support, this is a logical choice: by supporting the ATA standard this support can be reused for the memory module without any modification. However, in stand-alone devices, such as a SSA player, the application is burdened with file system details, if the memory module employs the block device approach. Therefore, stand-alone (portable) applications which require storage of multimedia content, may be built more efficiently if a controller unit on the memory module takes care of the file system details.

Figure 3 represents a schematic diagram of a memory module embodiment 20. For simplicity, electromechanical interfacing to the player has not been detailed in the Figure. The storage area 30 has an access time that is substantially independent of the physical storage location. The controller 22 controls the access to the storage proper. Various sub-systems have been shown therein, the host interface 24, the memory interface 26, and the file system 28. External write and internal selection to the memory are shown as well. Within the Application Programming Interface API the following functionality should be present. For memory formatting, an optional volume number is outputted that is either uniquely fixed and hard-wired, or a random number that is generated each time the command is executed. This number may only be changed when executing the formatting command, thereby destroying all data on the device. The copy protection proper does not expressly need this number. To create a file, a reusable file ID is produced for later referencing the file in question. When writing a block, a sector number is produced that is a random choice from the free block list. Depending on the implementation, the sector number that is produced can be the actual sector number in which the data proper were stored during the write operation, or it can be the sector number will be stored during the *next* write operation. This amongst others is possible in solid state audio devices without appreciable loss of time because the flash memory is not hampered by a seek time as is common in disk based systems. Such random choice in addition helps to level wear over the entire device. The application may use or discard the sector number returned by the block write command as required. When reading a block, the

file ID controls outputting the data proper and the sector number of the current or next block to be read.

Figure 4 illustrates an example of a file structure, that is distributed into blocks, each having the size of a single sector of 512 bytes. The first block carries information regarding the file, while the others have the file data proper. The above organization will block the making of a bit wise copy of the module, inasmuch as no modification facility for individual sectors has been provided. Copying to an intermediate storage location and subsequently recopying the data on the module (which constitutes the 'replay attack' as explained above) will copy the data to completely different locations. This in itself provides some protection against copying. Copy protection is further provided by encrypting a data block through a key that is derived from a secret and also from the location (for instance and preferably the sector number) where the data in question is stored. The latter information may be derived from the block write function that returns the sector number of the next file sector. As this information is not available for the first block, the latter may be used for less sensitive data. This limitation is overcome by letting the file create function return the sector number of the first sector in the file where the data proper (e.g. the file info) will be written. For reading, the present or next sector number is available before reading of the actual data, allowing the application to calculate the proper decryption key in time. The encryption key thus combines the location of the storage, and a method that renders it impossible to predict this location. Copying will change the storage location, and in consequence will break the relation between location and decryption key. Note that the secret used in the derivation of the key may be a globally shared secret between all players, or may be derived by other methods well known to those skilled in the art.

Figures 5A and 5B illustrate a method in accordance with the invention. Each time data blocks are written, the controller 22 writes the data in randomly chosen locations. In Figures 5A and 5B the locations are indicated by $L_1$, $L_2$ etc. The data are encrypted with a key which is dependent on a secret S and a location $L_i$ or a combination of locations $L_i$ (for instance the location of the block that is written, or of the previous block, or of the block that is written and the previous block etc).

Making a copy of the data of the memory module will (see Figure 5B) change in a irreproducible manner the locations of the data. In fact such will happen twice. Therefore, a recopy of a copy will have data for which the positions ($L_1$'', $L_2$'' etc.) does not correspond to the arguments needed for a proper decryption of the data. Subsequently the copy of the copy cannot be decrypted and is useless. The 'replay attack' is prevented.

Figures 6A and 6B show an embodiment of the invention in which all data are encrypted with a key K (which may consist of a single key or a block of keys), that itself is stored encrypted with a key K', which is the output of a hash function having as arguments the locations $L_1$, $L_2$, $L_3$ etc and a secret S. K' thus depends on the positions of the data blocks, in this case on the total sequence in which the data blocks are written. Since at each write access the locations $L_1$, $L_2$, $L_3$ are changed in an unpredictable manner, the result of the hash function H and thereby the key K' is changed. If the content is copied and recopied the player will fail (as in the method illustrated in Figures 5A and 5B) to recover the keys because K' is changed in an intractable manner. Accordingly any replay attack fails. Thus copying is prevented in an inexpensive manner requiring only moderate processing facilities and without the need of a unique identification code. It is noted that the invention provides the possibility of copy protection without the need for a unique identification code. This does not exclude use of such a code for other reasons or for extra protection.

It is also possible to arrange the data in groups of blocks, and groups of blocks are written in random locations. The same schemes as above may be used for groups of blocks, instead of single blocks. 'Random locations' within the concept of the invention in its broadest sense means locations that for all due purposes cannot be predicted in advance. 'For all due purposes' is stated since to get random numbers or locations use is usually made of some kind of algorithm. Substantially truly random, i.e. substantially evenly distributed throughout the memory module is preferred to even out wear on the device. Although preferably the method is applied to all or substantially all data in the memory module, the invention encompasses embodiments in which the method is applied to only a part of the data in the memory module. This could for instance be advantageous from the point of view of speed of operation. The invention is not restricted to using one and only one encryption method. When the data are divided in groups, embodiments using different encryption methods and different ways of dependency of said encryption methods on the locations may be used for different groups. This reduces the risk of unauthorized decryption. Although the controller may be provided in the system apart from the memory module, preferably the controller unit by which the random locations are chosen is integrated in the memory module. This makes it difficult to circumvent the method or influence the choice of locations of data.

In a method for providing copy-protection services on storage media, the locations where the data, preferably arranged in blocks, are stored, are chosen by a

(preferably built-in) controller on a random basis. Using an encryption key which depends critically on the position of the data the storage medium, decrypting copied data is made virtually impossible.